Herrnhuter Stern

Beim weihnachtlichen Stöbern im Internet stoße ich auf einen Artikel von J. Böhm[1], der ein Derive-Programm zum Zeichnen des Herrnhuter Sterns beschreibt. Für mich ein Anlass, dasselbe mit Maple zu versuchen, der Software, die ich als Mathematik-Programm benutze. Dieser Versuch dauerte länger als geplant – ich hatte lange nicht mehr mit dem Maple-System gearbeitet und vieles vergessen.

Zunächst lernte ich, dass der Rumpfkörper des Sterns ein archimedischer Körper ist, also ein konvexer Vielflächner, dessen Seitenflächen regelmäßige Vielecke sind und dessen Ecken sich zueinander gleich verhalten. Der Rumpfkörper des Herrnhuter Sterns ist ein Rhombenkuboktaeder. Seine Oberfläche besteht aus 18 Quadraten und 8 gleichseitigen Dreiecken. Jeweils drei Quadrate und ein Dreieck bilden eine Raumecke, und jeweils 8 Kanten sind Kanten eines regelmäßigen Achtecks. Insgesamt gibt es sechs dieser Achtecke. Abbildung (1) zeigt ein mit Maple gezeichnetes Drahtgittermodell eines Rhombenkuboktaeders. Die Koordinaten der Eckpunkte der sechs Achtecke wurden der Arbeit von Böhm[1] entnommen.

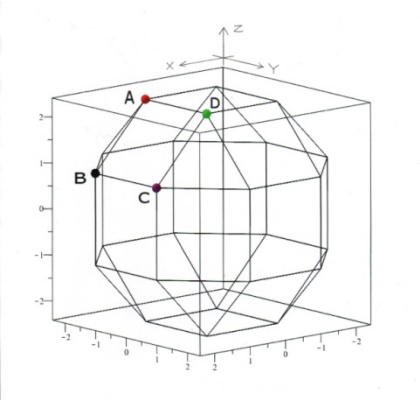


Abbildung 1 Rhombenkuboktaeder. Die gekennzeichneten Eckpunkte werden weiter unten betrachtet, siehe Text.

Die Zacken des Sterns entstehen, indem man auf jeder Seitenfläche des Rumpfkörpers eine gerade Pyramide errichtet, insgesamt also 18 quadratische Pyramiden und 8 Pyramiden mit der Grundfläche eines gleichseitigen Dreiecks.

Zum Zeichnen der Pyramiden benötigt man die Koordinaten der Eckpunkte des Rumpfkörpers – sie bilden die 18 Quadrate und 8 Dreiecke der Pyramiden-Grundfläche – und die Koordinaten der jeweiligen Pyramidenspitze. Da der Rhombenkuboktaeder zu jeder seiner Mittelachsen drehsymmetrisch ist, genügt es, die Eckpunktkoordinaten einer Seitenfläche zu ermitteln und diese Koordinaten einer Drehung um die passende Achse zu unterwerfen. In unserem Beispiel benutzen wir die in Abbildung (1) markierten Eckpunkte *A*, *B*, *C* und *D* der in *x*-Richtung schräg nach oben zeigenden Seitenfläche. Wir drehen die Koordinaten dieser Punkte um die in *z*-Richtung zeigende Mittelachse des Rhombenkuboktaeders, und erhalten so die Eckpunktkoordinaten der übrigen drei Seitenflächen, deren Normalen ebenfalls unter 45° nach oben geneigt sind.

Der Ursprung des Koordinatensystems sei der Schnittpunkt der Mittelachsen. Die Kantenlänge der Seitenflächen wählen wir zu $a = 2$. Dann haben die vier Ecken des Quadrats *ABCD* in Abbildung (1) die Koordinaten $A(1, -1, 1+\sqrt{2})$, $B(1+\sqrt{2}, -1, 1)$, $C(1+\sqrt{2}, 1, 1)$ und $D(1, 1, 1+\sqrt{2})$. Diese Punkte erzeugen durch Drehung um die *z*-Achse mit den Winkeln 90°, 180° und 270° die Ecken der anderen unter 45° geneigten Seitenflächen. Die Drehmatrix für eine Drehung um die *z*-Achse lautet

$$(1) \qquad M_z(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

Eine Drehung um $\phi = 90°$ entspricht also

$$(2) \qquad M_z(90°) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

so dass die Ecke A' der Seitenfläche mit der Normalen in y-Richtung sich zu

$$(3) \qquad A' = M_z(90°) \cdot A = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ -1 \\ 1+\sqrt{2} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1+\sqrt{2} \end{bmatrix}$$

ergibt. In gleicher Weise folgen die Eckpunkte $B'(1, 1+\sqrt{2}, 1)$, $C'(-1, 1+\sqrt{2}, 1)$ und $D'(-1, 1+\sqrt{2}, 1)$. Die weiteren Seitenflächen mit der Normalen unter + 45° ergeben sich für ϕ = 180° und 270°.

Die zwei Seitenflächen mit $z = \pm (1+\sqrt{2})$ brauchen natürlich nicht gedreht zu werden. Für die acht Seitenflächen mit waagerechter Normalen (xy-Richtung) sind ϕ = 45°, 90°, 135°, … zu setzen.

Um die Koordinaten der jeweiligen Pyramidenspitze zu erzeugen, errichten wir über dem Mittelpunkt der Seitenfläche die Flächennormale. Diese Gerade bringen wir zum Schnitt mit einer Kugel von frei wählbarem Radius.

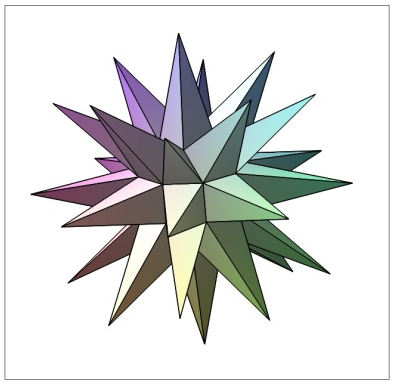


Abbildung 2 Mit dem Maple-Programm gezeichneter Herrnhuter Stern.

Abbildung 3 Herrnhuter Stern

Abbildung 2 zeigt das Ergebnis meines Zeichenprogramms, das Foto Abbildung 3 einen beleuchteten Herrnhuter Stern.

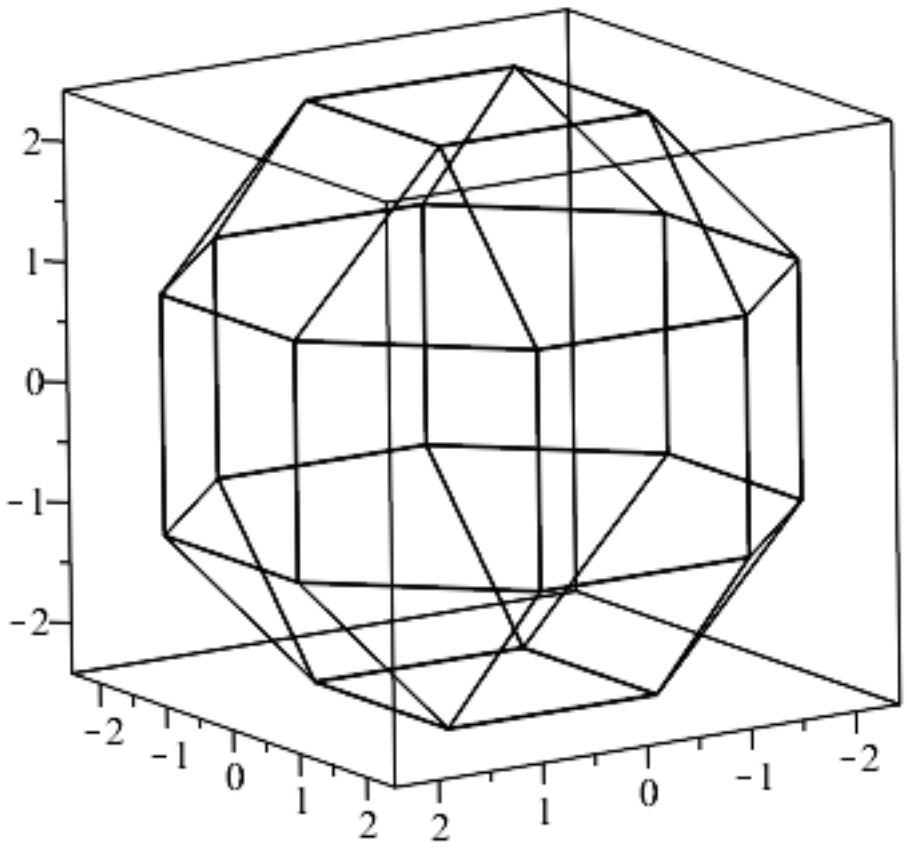
[1] Josef Böhm, *Der Herrnhuter Stern*, nojo.boehm@pgv.at

```
> restart: with(geom3d): with(plots): with(LinearAlgebra):
```

Zeichnen der sechs Achtecke des Drahtgittermodells des Rhombokuboktaeders

```
> r := sqrt(2*sqrt(2)+4):
> t := 2.42:
> Achteckm1 := evalf(Matrix([seq([r*cos((Pi/4)*i + Pi/8), r*sin(
  (Pi/4)*i + Pi/8), -1], i=0..8)])):
> m1 := pointplot3d(Achteckm1,view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt
  (2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)], color=black, style=line,
  axes=boxed):
> Achteckm2 := evalf(Matrix([seq([r*cos((Pi/4)*i + Pi/8), r*sin(
  (Pi/4)*i + Pi/8), 1], i=0..8)])):
> m2 := pointplot3d(Achteckm2, view=[-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)],color=black, style=
  line):
> Achtecks1 := evalf(Matrix([seq([-1, r*cos((Pi/4)*i + Pi/8), r*
  sin((Pi/4)*i + Pi/8)], i=0..8)])):
> s1 := pointplot3d(Achtecks1,view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt
  (2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)],color=black, style=line):
> Achtecks2 := evalf(Matrix([seq([1, r*cos((Pi/4)*i + Pi/8), r*
  sin((Pi/4)*i + Pi/8)], i=0..8)])):
> s2 := pointplot3d(Achtecks2, view=[-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)], color=black, style=
  line):
> Achteckq1 := evalf(Matrix([seq([r*cos((Pi/4)*i + Pi/8), -1, r*
  sin((Pi/4)*i + Pi/8)], i=0..8)])):
> P := pointplot3d({[r*cos((Pi/4)*1 + Pi/8), -1, r*sin((Pi/4)*1 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=red, symbol=solidcircle, symbolsize=
  20):
> Q:= pointplot3d({[r*cos((Pi/4)*2 + Pi/8), -1, r*sin((Pi/4)*2 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=blue, symbol=solidcircle,
  symbolsize=20):
> U:= pointplot3d({[r*cos((Pi/4)*8 + Pi/8), -1, r*sin((Pi/4)*8 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=black, symbol=solidcircle,
  symbolsize=20):
> q1 := pointplot3d(Achteckq1,view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt
  (2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)], color=black, style=line)
  :
> Achteckq2 := evalf(Matrix([seq([r*cos((Pi/4)*i + Pi/8), 1, r*
  sin((Pi/4)*i + Pi/8)], i=0..8)])):
> R:= pointplot3d({[r*cos((Pi/4)*1 + Pi/8), 1, r*sin((Pi/4)*1 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=green,symbol=solidcircle,
  symbolsize=20):
> S:= pointplot3d({[r*cos((Pi/4)*2 + Pi/8), 1, r*sin((Pi/4)*2 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=orange, symbol=solidcircle,
  symbolsize=20):
> V:= pointplot3d({[r*cos((Pi/4)*8 + Pi/8), 1, r*sin((Pi/4)*8 +
  Pi/8)]}, view=[-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2)], color=maroon, symbol=solidcircle,
  symbolsize=20):
```

```
> q2 := pointplot3d(Achteckq2, view=[-1-sqrt(2)..1+sqrt(2),-1-
  sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)], color=black, style=
  line):
> W := pointplot3d({[1,1+sqrt(2),1]}, view=[-1-sqrt(2)..1+sqrt
  (2),-1-sqrt(2)..1+sqrt(2),-1-sqrt(2)..1+sqrt(2)], color=maroon,
  symbol=solidcircle, symbolsize=20):
> display(m1, m2, s1, s2, q1, q2, view=[-t..t,-t..t,-t..t], axes=
  boxed);
```




```
> #display(P, Q, R, S, U, V, W):
```

Pyramiden auf quadratischen Grundflächen

```
> MZ := phi -> Matrix([[cos(phi),-sin(phi),0],[sin(phi),cos(phi),
  0],[0,0,1]]):

> t := 8:

> pyrSq := proc(A,B,C,D, pyr)
    local E, AC, M, nE, s, Is, Sh, tp:
    plane(E, [A,B,C]):
    Equation(E, [x,y,z]):
    segment(AC, [A,C]):
    midpoint(M, AC):
    line(nE, [M, vector(NormalVector(E))]):
```

```
    Equation(nE, 't'):
    sphere(s,x^2+y^2+z^2=8^2,[x,y,z]):
    intersection(Is, nE, s):
    point(Sh, coordinates(Is[1])):
    triangle(t1, [A,B,Sh]):
    triangle(t2, [B,C,Sh]):
    triangle(t3, [C,D,Sh]):
    triangle(t4, [D,A,Sh]):
    pyr := draw([t1,t2,t3,t4]):
    #display([m1, m2, s1, s2, q1, q2, pyr], view=[-t..t,-t..t,-t.
  .t], orientation=[65,35], axes=boxed);
  end proc:
```

```
> drehung := proc(phi, A0,B0,C0,D0)
    local Av, Bv, Cv, Dv:
    Av := Vector(coordinates(A0)):
    Bv := Vector(coordinates(B0)):
    Cv := Vector(coordinates(C0)):
    Dv := Vector(coordinates(D0)):
    point(A, convert(MZ(phi).Av,list)):
    point(B, convert(MZ(phi).Bv,list)):
    point(C, convert(MZ(phi).Cv,list)):
    point(D, convert(MZ(phi).Dv,list)):
  end proc:
```

```
> point(A0,[1,-1,1+sqrt(2)]):
> point(B0,[1,1,1+sqrt(2)]):
> point(C0,[-1,1,1+sqrt(2)]):
> point(D0,[-1,-1,1+sqrt(2)]):
```

```
> pyrSq(A0,B0,C0,D0, 'pyr'):
intersection: two points of intersection
```

```
> pyram[1] := pyr:
```

```
> display([m1, m2, s1, s2, q1, q2, pyr], view=[-t..t,-t..t,-t..
  t], orientation=[65,35], axes=boxed):
```

```
> point(A0 , [1,-1,1+sqrt(2)]):
> point(B0 , [1+sqrt(2),-1,1]):
> point(C0 , [1+sqrt(2),1,1]):
> point(D0 , [1,1,1+sqrt(2)]):
```

```
> for k from 0 to 3 do
    drehung(k*Pi/2,A0,B0,C0,D0):
    pyrSq(A,B,C,D,'pyr'):
    pyramide[k] := pyr:
  end do:
  display([m1, m2, s1, s2, q1, q2, seq(pyramide[k],k =0..3)],
  view=[-t..t,-t..t,-t..t], orientation=[65,35], axes=boxed):
Warning, a geometry object has been assigned to the protected
name D.  Use of protected names for geometry objects is not
recommended and may break Maple functionality.
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
```

```
intersection: two points of intersection
> pyram[2] := seq(pyramide[k],k =0..3):

> point(A0 , [1+sqrt(2),-1,1]):
> point(B0 , [1+sqrt(2),-1,-1]):
> point(C0 , [1+sqrt(2),1,-1]):
> point(D0 , [1+sqrt(2),1,1]):


> for k from 0 to 7 do
    drehung(k*Pi/4,A0,B0,C0,D0):
    pyrSq(A,B,C,D,'pyr'):
    pyramide[k] := pyr:
  end do:
  display([m1, m2, s1, s2, q1, q2, seq(pyramide[k], k=0..7)],
  view=[-t..t,-t..t,-t..t], orientation=[65,35], axes=boxed):
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
> pyram[3] := seq(pyramide[k],k =0..7):

> point(A0 , [1+sqrt(2),-1,-1]):
> point(B0 , [1,-1,-(1+sqrt(2))]):
> point(C0 , [1,1,-(1+sqrt(2))]):
> point(D0 , [1+sqrt(2),1,-1]):

> for k from 0 to 3 do
    drehung(k*Pi/2,A0,B0,C0,D0):
    pyrSq(A,B,C,D,'pyr'):
    pyramide[k] := pyr:
  end do:
  display([m1, m2, s1, s2, q1, q2, seq(pyramide[k],k=0..3)],
  view=[-t..t,-t..t,-t..t], orientation=[65,35], axes=boxed):
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection

> pyram[4] := seq(pyramide[k],k=0..3):

> point(A0,[1,-1,-(1+sqrt(2))]):
> point(B0,[-1,-1,-(1+sqrt(2))]):
> point(C0,[-1,1,-(1+sqrt(2))]):
> point(D0,[1,1,-(1+sqrt(2))]):

> pyrSq(A0,B0,C0,D0, 'pyr'):
intersection: two points of intersection
> display([m1, m2, s1, s2, q1, q2, pyr], view=[-t..t,-t..t,-t..
  t], orientation=[65,35], axes=boxed):
> pyram[5] :=  pyr:
```

Pyramiden auf dreieckigen Grundflächen

```
> point(A0, [1,1,1+sqrt(2)]):
> point(B0, [1+sqrt(2),1,1]):
> point(C0, [1,1+sqrt(2),1]):

> pyrDr := proc(A,B,C, pyr)
    local E, AC, BC, NAC, MBC, sAC, sBC, M, nE, s, Is, Sh, tp:
    plane(E, [A,B,C]);
    Equation(E, [x,y,z]);
    segment(BC, [B,C]);
    midpoint(MBC, BC);
    segment(AC, [A,C]);
    midpoint(MAC, AC);
    line(sBC, [A,MBC]);
    line(sAC, [B,MAC]);
    intersection(M, (sAC,sBC));
    coordinates(M);
    line(nE, [M, vector(NormalVector(E))]);
    Equation(nE, 't');
    sphere(s,x^2+y^2+z^2=8^2,[x,y,z]);
    intersection(Is, nE, s);
    point(Sh, coordinates(Is[1]));
    coordinates(Sh);
    triangle(t1, [A,B,Sh]);
    triangle(t2, [B,C,Sh]);
    triangle(t3, [C,A,Sh]);
    pyr := draw([t1,t2,t3]);
    #display([m1, m2, s1, s2, q1, q2, pyr], view=[-t..t,-t..t,-t.
  .t], orientation=[65,35], axes=boxed):
  end proc:

> drehungDr := proc(phi, A0,B0,C0)
    local Av, Bv, Cv:
    Av := Vector(coordinates(A0)):
    Bv := Vector(coordinates(B0)):
    Cv := Vector(coordinates(C0)):
    point(A, convert(MZ(phi).Av,list)):
    point(B, convert(MZ(phi).Bv,list)):
    point(C, convert(MZ(phi).Cv,list)):
  end proc:

> for k from 0 to 3 do
    drehungDr(k*Pi/2,A0,B0,C0):
    pyrDr(A,B,C, 'pyr'):
    pyramide[k] := pyr:
  end do:
  display([m1, m2, s1, s2, q1, q2, seq(pyramide[k],k =0..3)],
  view=[-t..t, -t..t,-t..t], orientation=[65,35], axes=boxed):
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
> pyram[6]:= seq(pyramide[k],k =0..3):

> point(A0, [1,1,-(1+sqrt(2))]):
> point(B0, [1,1+sqrt(2),-1]):
> point(C0, [1+sqrt(2),1,-1]):
```

```
> for k from 0 to 3 do
    drehungDr(k*Pi/2,A0,B0,C0):
    pyrDr(A,B,C, 'pyr'):
    pyramide[k] := pyr:
  end do:
  display([m1, m2, s1, s2, q1, q2, seq(pyramide[k],k =0..3)],
  view=[-t..t, -t..t,-t..t], orientation=[65,35], axes=boxed):
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
intersection: two points of intersection
> pyram[7] := seq(pyramide[k],k =0..3):

> display([m1, m2, s1, s2, q1, q2, pyram[1], pyram[2], pyram[3],
  pyram[4], pyram[5], pyram[6], pyram[7]], view=[-t..t,-t..t,-t..
  t]);
```